On the Analysis of Groupware Usability Using Annotated GOMS

PEDRO ANTUNES
MARCOS R.S. BORGES
JOSE A. PINO
LUÍS CARRIÇO

DI-FCUL TR–04–18

DEZEMBRO 2004

Departamento de Informática
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, 1749-016 Lisboa
Portugal

Technical reports are available at http://www.di.fc.ul.pt/tech-reports. The files are stored in PDF, with the report number as filename. Alternatively, reports are available by post from the above address.

On the Analysis of Groupware Usability Using Annotated GOMS

Pedro Antunes [1]

Marcos R. S. Borges [2]

Jose A. Pino [3]

Luís Carriço [4]

[1,4] *Department of Informatics, Faculty of Sciences of the University of Lisboa*

Edificio C6, Piso 3, Campo Grande,

1749-016 Lisboa, Portugal

[1]*paa@di.fc.ul.pt* [4]*lmc@di.fc.ul.pt*

[2] *Graduate Program in Informatics – NCE & IM – Federal University of Rio de Janeiro (UFRJ)*

Caixa Postal 2324,

20001-970 Rio de Janeiro, Brazil

mborges@nce.ufrj.br

[3] *Department of Computer Science – Universidad de Chile*

Blanco Encalada 2120, Piso 3,

Santiago 6511224, Chile

jpino@dcc.uchile.cl

Abstract

GOMS is a well-known model that has been successfully used in predicting the performance of human-computer interaction, identifying usability problems and improving user-interface design. The focus of GOMS is on the individual user, however. This explains why it has no significant impact in the groupware context. This paper discusses the applicability of GOMS in the groupware context. We analyzed the impact of groupware in the cognitive architecture of GOMS in order to

accomplish this goal. The obtained results led us to introduce an annotation scheme in GOMS reflecting several pre and post conditions necessary to describe how operations realized by collaborating users are interrelated in GOMS descriptions. We discuss the applicability of Annotated GOMS by studying a collaborative tool for software engineering requirements negotiation. This work contributes to the collaboration systems field with an innovative way to analyze groupware usability.

Keywords: GOMS, Groupware Usability, Groupware Cognitive Architecture.

1 Introduction

Collaborative systems place many challenges to usability evaluation (Ivory & Hearst, 2001), motivated by the number of users required to participate in the evaluation process and the required control over many technological factors and other variables related to the group, task and context (see, e.g., (Fjermestad & Hiltz, 1999)). The complexity and cost of usability evaluation may be impeding the emergence of the best groupware designs, highly usable and useful to individuals, work groups and organizations.

A collection of discount methods has recently emerged with the purpose of reducing the complexity and cost of groupware usability evaluation (Baker, et al., 2002). Many of these methods resulted from the adaptation of discount methods used to evaluate single-user software (*singleware*), such as groupware heuristic evaluation (Baker, et al., 2002), groupware usability inspection (Steves, et al., 2001), groupware walkthrough (Pinelle & Gutwin, 2002) and scenario based evaluation (Haynes, et al., 2004).

GOMS (Goals, Operations, Methods and Selection Rules (Card, et al., 1983)) and its family of models, such as GOMSL (e.g., (Kieras, 1999)) also fall in the category of discount methods for singleware evaluation, by providing an analytic

approach that can be applied without the participation of users and even without a prototype being developed (John, 1995). This approach has been successfully brought to operation in several ways, to predict usability, optimize user interaction or benchmark various design solutions (John & Kieras, 1996). GOMS can also be used to make predictions about human costs of using systems or training users (John, 1995).

GOMS addresses singleware interactions, i.e. one user interacting with one device (John, 1995). It is possible to model multiple user interactions with one device using GOMS, as reported by Kieras and Santoro (2004). However, we realized that such an approach is limited to coordination support and not beneficial for groupware designers, since the focus is on an individual basis whereas groupware designers are mostly interested in the collaborative context.

We argue the GOMS approach may also be added to the existing collection of discount groupware usability evaluation methods, offering additional contributions to groupware design that are not covered by the other methods. The potential advantages of this approach emerge from some fundamental characteristics of GOMS. We would like to emphasize the following ones:

- One important argument in favor of using GOMS is that it affords studying the usability of alternative design solutions in an analytical way (Kieras et al., 1997). This approach may save design time and effort by reducing the number of iterations and empirical tests necessary to revise and improve an initial design (Khalifa, 1998).
- Another important feature of GOMS is that it is founded on a cognitive architecture providing insights about the assumed mechanisms and

capabilities of the human processing system (Kieras, 1999). These insights may be instrumental to designers aiming to develop good groupware tools.

- GOMS is applied to situations where users accomplish tasks that they already master (John, 1995). This excludes using GOMS to analyze exploratory and creativity scenarios, decision making processes, as well as situations where users must learn how to use the system. Although the scope is apparently more limited when compared to the other approaches, we note that it may be used to analyze the fine-grained details of collaboration required, for instance, when using shared workspaces for intensive collaboration.
- GOMS offers an engineering solution with quantitative estimates of human performance (John & Kieras, 1996). We conjecture some of these quantitative estimates may be extrapolated to groupware interaction.

Recognizing the strong theoretical and practical foundations of GOMS, we were interested in studying the applicability of GOMS to the collaborative context. We restricted the study to the specific context of concerted work, i.e. people working together in a concerted effort towards a shared goal (Nunamaker, et al., 1997); and to the situation where work is exclusively accomplished through groupware support.

The GOMS cognitive architecture (user and device) and building blocks (goals, operators, methods and selections rules) approximate human-computer interaction at a low level of detail. We investigated which specializations could be made in the cognitive architecture and building blocks to reflect the particular characteristics of groupware in GOMS. This endeavor is explained in Section 3. Then, we defined an annotation scheme to describe how operations realized by collaborating users are interrelated in GOMS descriptions. This annotation scheme is explained in

Section 4. We proceed with an in-depth analysis of a groupware tool to demonstrate the applicability of the annotation scheme in Section 5. Specifically, we analyze a collaborative tool for software engineering requirements negotiation and discuss the insights that Annotated GOMS brings to groupware designers. We discuss the benefits and limitations of the approach in Section 6. Section 7 contains the conclusions of the research.

2 Related Work

We begin with an overview of several discount usability methods specifically developed for groupware. The first one is groupware walkthrough (Pinelle & Gutwin, 2002), a method based on cognitive walkthrough (Polson, et al., 1992). The approach starts with a task description using a set of small-scale group activities named "mechanics of collaboration" (Steves, et al., 2001). When these task descriptions are available, a set of expert evaluators review the tasks and analyze how the shared workspace supports the users' goals. The major adaptations of cognitive walkthrough to the groupware context resulted from the observation that single-user actions should be filtered out from task descriptions, while groupware walkthrough should model multiple concurrent tasks and multiple choices to accomplish typical tasks in collaborative work (Pinelle & Gutwin, 2002).

Another discount usability method is heuristic evaluation (Baker, et al., 2002). This method, adapted from the heuristic evaluation methodology (Nielsen, 1992), is based on a set of experts evaluating the compliance of a shared workspace with a list of heuristics. As with the groupware walkthrough approach, the list of heuristics is founded on the mechanics of collaboration.

Considering that both heuristic evaluation and groupware walkthrough are dependent on the quality of the task analysis, Pinelle et al. (2003) proposed CUA

(Collaboration Usability Analysis), an improved version of the mechanics of collaboration. It is interesting to compare CUA with GOMS. Like GOMS, CUA analyzes tasks using a hierarchical model. An important difference between these two approaches lies in the level of task decomposition. The CUA lowest granularity defines mechanical collaborative actions that users perform in shared workspaces, such as writing a message or obtaining a resource from the shared workspace. GOMS decomposes tasks at a much lower level of detail, e.g., single keystrokes. However, this level of detail afforded by GOMS is unrelated to collaboration, and increasing the level will naturally approximate both approaches in terms of the insights they may provide to groupware designers.

Furthermore, we argue there are additional differences that may justify applying GOMS to the groupware context. One of them is that GOMS is founded on a cognitive architecture that has no counterpart in CUA. We hypothesize that a cognitive architecture adapted to groupware may provide additional insights about how users interact with groupware. In particular, GOMS goes beyond external actions (what users do) and also address internal actions (what users think), which may provide additional insights about task conditions involved in collaborative work. Focusing on task conditions affords designers to think about how to develop shared artifacts that allow users to easily grasp the design logic behind groupware.

Another difference to ponder is that GOMS addresses tasks which are well-known and mastered by the users (John, 1995). We hypothesize the design of intensive collaborative tools - where the designer may find necessary to optimize the effort applied by users in low-level collaborative activities - may benefit from the GOMS approach.

Scenario based evaluation (Haynes et al., 2004) is another groupware discount usability method. It was derived from scenario based design (Carrol, 2000) and consists of collecting detailed narratives of users' interactions with a system when performing a specific task. Each scenario identifies an actor, setting, task goals and claims. The claims are subsequently analyzed and related to system features in order to identify their positive and negative aspects. We regard this approach as unrelated to GOMS, since it does not offer an analytical approach.

Groupware task analysis (van der Veer & van Welie, 2000) is a method intended to generate task descriptions of current and future situations involving groupware use. These task descriptions are similar to the work models defined by the contextual design approach (Beyer & Holtzblatt, 1998), well know in the HCI field. Although a usability framework is proposed to afford early evaluation of groupware designs, this method is more oriented towards design than to evaluation.

We will now overview the application of GOMS in the groupware context. Min et al. (1999) developed DGOMS (Distributed GOMS) as an extension of GOMS to the group level of analysis. The approach regards group work at a high level of detail, as a group task that can be successively decomposed in group subtasks until individual tasks can be identified. A new type of operator, called communication operator, is defined to coordinate individual tasks executed in parallel. Therefore, this approach does not address concerted but coordinated work. As mentioned above, we focus our study on concerted work in this paper.

Kieras and Santoro (2004) applied GOMS to a complex task executed by a team of users. The task involved several users with individual roles monitoring a display and executing actions in a coordinated way. Coordination was supported by a

shared radio communication channel. As in the previous case, this approach does not address concerted work.

3 The GOMS Architectural Basis and its Relation to Groupware

In general, the GOMS family of models has been associated with the Human Processor Model (Card, et al., 1983), which represents human information processing capabilities using perceptual, motor and cognitive processors. However, significant architectural differences are identified when considering individual models. For instance, KLM (Card, et al., 1980) uses a serial-stage architecture, while EPIC (Kieras, et al., 1997) addresses multimodal and parallel human activities. In spite of these differences, a common characteristic to the whole GOMS family of models is that it is singleware (Ritter, et al., 2000), i.e., it assumes that one single user interacts with a physical interface comprising several input and output devices.

Figure 1 depicts this singleware architecture based on the EPIC architecture (some components considered not relevant to our purposes, like the production memory, are absent). We also illustrate that there is one conventional flow of information in the architecture (Kieras, et al., 1997), from the cognitive processor to the motor processors, input devices, output devices, perceptual processors and then back to the cognitive processor.

According to some authors (Kieras & Santoro, 2004), the architecture depicted in Figure 1 applies to groupware in a very transparent way: in order to model a team of users, one can model each individual interaction between the user and the physical interface; and assume that (1) the physical interface is shared by multiple users and (2) the users will deploy procedures and strategies to communicate and coordinate their individual actions. Thus, according to this view, groupware usage will be reflected in some conventional flows of information, spanning several users, which

still may be described using the conventional production rules and representations. Kieras and Santoro (2004) studied the performance of a team of operators connected via speech over an intercom channel in this way.

The problem, however, is that this approach does not reflect two fundamental issues with groupware: (1) the focus and granularity should not remain on the interactions between user and physical interface but should significantly change to focus on the interactions between users, mediated by the physical interface; and (2) with groupware, the conventional flows of information are considerably changed to reflect concerted work. From our point of view, we have to re-analyze the users' cognitive processing of the conventional flows of information in order to address these groupware issues. We must also discuss these flows in relation to multi-user interactions.

Let us start with the singleware architecture. In that context, we may characterize the conventional flow of information in two categories: feedback and feedforward. The first category corresponds to a flow of information initiated by the user, for which the physical interface conveys *feedback* information to make the user aware of the executed operations (Douglas & Kirkpatrick, 1999; Wensveen, et al., 2004). The second category concerns the delivery of *feedforward* information, initiated by the physical interface, to make the user aware of the afforded action possibilities.

Now, when we regard groupware, some additional categories may have to be considered. We will analyze three different categories: explicit communication, feedthrough and back-channel feedback. The *explicit communication* category, as defined by Pinelle et al. (2003), addresses information produced by one user and explicitly intended to be received by other users. This situation can be modeled as a

physical device capable to multiplex information from input devices to several output devices (Kieras & Santoro, 2004). The immediate impact on the model shown in Figure 1 is that we now have to explicitly consider additional users connected to the physical device.

The *feedthrough* category concerns implicit information delivered to several users reporting actions executed by one user (Hill & Gutwin, 2003). This flow of information is initiated by the physical interface and is directed towards the other users. A simple form of generating feedthrough information consists of multiplexing feedback information to several users. Sophisticated schemes may consider delivering less information by manipulating the granularity and timing associated to the operations executed by the groupware system (Gutwin & Greenberg, 1999).

The notion of feedthrough has an important impact on task modeling for several reasons. The first one is that feedthrough is essential to provide awareness about the other users and construct a context for collaboration. We can regard the processing of awareness information in a specialized perceptual processor, capable of processing sensory information about who, what, when, how, where are the other users operating in the system. We will call *awareness processor* to this specialized perceptual processor. We may also model the delivery of feedthrough to the awareness processor using a specialized output device, which we will name *awareness output device*. Another feature of the awareness processor is that it does not only afford users to construct a perceptual image of the collaborative context, but it allows users to perceive the role and limitations of the physical interface as a mediator. This is particularly relevant when Internet is being used to convey feedthrough, causing feedthrough delays which are significantly higher and unpredictable than feedback delays (Gutwin, et al., 2004).

An additional reason for analyzing the impact of feedthrough on task modeling is related to another particular characteristic of groupware: it affords users to loose the link between executed operations and awareness – a situation that is called loosely coupled (Dewan & Choudhary, 1995). Two types of control are generally supported by groupware in a loosely coupled situation: (1) the user may get awareness information on a per-object demand basis, e.g. by moving the focus of interest; or (2) the user specifies filters that restrict awareness to some selected objects and types of events. In both cases this situation requires some cognitive activities from the user to discriminate and control awareness information, which can be modeled as one specialized motor processor, called *coupling processor*. A specialized input device will be devoted to control awareness information in the physical interface.

Finally, the *back-channel feedback* category concerns information flows initiated by one user and directed towards another user to facilitate communication (Rajan, et al., 2001). No significant content is delivered through back-channel feedback, because it does not reflect cogitation from the user. That is the main difference between explicit communication and back-channel feedback. We can model this type of activity as a motor activity executed by the coupling processor in response to some perceived inputs. The outputs produced by this motor activity will be delivered to other users through the awareness output device.

In Figure 2 we illustrate the obtained groupware architecture. In summary, our interpretation of the GOMS architecture, taking the groupware context in consideration, essentially consists of introducing the awareness processor and output device, handling awareness information from the other users operating in the system; and the coupling processor and input device, responsible for controlling the amount of awareness information delivered to one user.

Observe that this groupware architecture does not imply any modifications to the GOMS architecture, providing instead a contextualization adequate to a specialized application area, namely groupware. Finally, note this groupware architecture does not address face-to-face situations where users, besides collaborating though groupware, may exploit other verbal, visual and body communication channels.

4 Modeling Groupware - Annotated GOMS

This section discusses how to model groupware using the proposed architecture. We start the discussion with a canonical example where user A moves an object to a shared workspace and user B moves that object to another destination. The following model describes this goal using NGOMSL (Kieras, 1996) and the singleware architecture.

```
Method for goal: Move object between users.
Step 1. Accomplish goal: Move object to shared space.
Step 2. Locate object on screen.
Step 3. Accomplish goal: Move object to private space.
Step 4. Return with goal accomplished.

Method for goal: Move object to destination.
Step 1. Accomplish goal: Pick object.
Step 2. Point to destination.
Step 3. Accomplish goal: Release object.
Step 4. Return with goal accomplished.

Method for goal: Pick object.
Step 1. Determine position of object.
Step 2. Point to object.
Step 3. Hold down mouse button.
Step 4. Return with goal accomplished.

Method for goal: Release object.
Step 1. Verify that object is at destination.
Step 2. Release mouse button.
Step 3. Return with goal accomplished.
```

Observe that step 1 in the “move object between users” method refers to a goal of user A, while steps 2 and 3 refer to user B. Furthermore, step 2 in the “pick object” method is associated to feedforward, while the step 1 in the “release object” method is related to feedback.

We will however focus on feedthrough to discuss this example. Feedthrough information is fundamentally associated to step 2 of the "move object between users" method, where user B becomes aware that the object from user A was moved to the shared workspace. Unfortunately, the description of step 2 does not convey any explanation about what type of operation generated feedthrough, something that, from a groupware point of view, is needed to fully understand the interaction between users A and B. Furthermore, depending on the implementation of the feedthrough mechanism, the physical device may simply produce outputs when the object is released on the shared workspace, or, alternatively, produce intermediate outputs while the object is being dragged by the pointer (as in a telepointer (Dyck, et al., 2004)). Moreover, this type of information cannot be represented with additional operators, since it is related to the device functionality and not to the user. Therefore, we propose to add the notion of pre and post conditions to method descriptions to resolve these problems. This is shown below in the redesigned "move object between users" method.

```
Method for goal: Move object between users.
Step 1. Move object to shared space. (A)
Step 2. Move object from shared space. (B)
Step 3 Return with goal accomplished.

Method for goal: Move object to shared space. (A)
Step 1. Accomplish goal: Move object to destination. → object
      released
Step 2. Return with goal accomplished.

Method for goal: Move object from shared space. (B)
Step 1. Locate object on screen. ← object released
Step 2. Accomplish goal: Move object to destination.
Step 3. Return with goal accomplished.
```

The "move object to shared space" method has now a post-condition annotated in step 1, denoted by the → symbol, which states that feedthrough is produced as a consequence of the action executed in that step. The post-condition also refers to the type of feedthrough produced, e.g. in this case the information that an object was released in the shared workspace. The "move object from shared space" method has

been annotated with a pre-condition, denoted by the ← symbol. This pre-condition states that the accomplishment of step 1 by the user involves processing the corresponding feedthrough information.

The basic feature of pre and post conditions is they are notational mechanisms, helping the designer identifying the context of collaboration without interfering with the GOMS description. Of course, this approach affords various levels of detail. For instance, in this canonical example we have not described the details of the "locate object on screen" method. One design alternative could involve user B waiting and then processing the received feedthrough information with the awareness processor. Another design alternative could involve a shared workspace operated in a loosely coupled way, where user B would obtain feedthrough selectively, by moving a viewport over the shared workspace.

The following description affords that functionality for user B. Once again, we use a post condition to establish a relation of causality between moving the viewport to a specific region and receiving feedthrough.

```
Method for goal: Move focus on shared space. (B)
Step 1. Adjust coupling device. → [area]
Step 2. Return with goal accomplished.

Method for goal: Move object from shared space. (B)
Step 1. Locate object on screen. ← [area] object released
Step 2. Accomplish goal: Move object to destination.
Step 3. Return with goal accomplished.
```

We would like to point out that while a high level of detail affords the designer studying how users establish collaboration patterns, the low level of detail also affords studying in detail and optimizing the support provided by shared workspaces. This optimization may in fact be a central issue when developing groupware for intensive collaboration.

Finally, we observe the "move object to destination" method, as well as the other methods defined below that one, are irrelevant to explain the collaboration

between A and B in this specification. As we previously mentioned, this is because feedback and feedforward are associated to singleware but not to groupware.

In summary, in this example we were able to describe the collaboration between users A and B by applying a very simple annotation scheme to NGOMSL: using pre and post conditions to describe how the operations realized by users A and B are interrelated. We will now discuss in detail the syntax of pre and post conditions shown in Table 1.

We organize pre and post conditions in three major categories: feedthrough, back-channel feedback and coupling. The first category includes conditions associated to basic awareness information about the shared workspace (who is in the group, what, when, how and where are they), object manipulations (sharing, acquiring, releasing, etc); and mouse, key or text events (affording more fine grained awareness information than object manipulations). The second category considers three types of prompts used to keep conversations on track: auditory, verbal and visual. Finally, the third category addresses the user control over feedthrough delivery, which may be coupled to selected areas of the shared workspace, selected objects, events or time periods.

Finally, we may also specify several operators introduced by the groupware perspective (shown in Table 2). These are not new operators in terms of what is specified for DGOMSL, but are rather specializations of existing operators, something that is natural considering that both the awareness and coupling processors are specializations of perceptual and motor processors as well. For instance, related to the awareness processor, we specialize the verify operator to the different types of feedthrough being delivered to users. Concerning the coupling processor, we added a specialization of the "move cursor" and "point to" primitive operators, called move

focus, which encapsulates the set of operations necessary to control coupling, e.g. moving a viewport. The prompt operator, which is also a specialization of the primitive motor operators, is dedicated to produce back-channel information necessary to control the dialogue between users.

5 Case Study

The following case study is intended to discuss in detail the approach developed above. The case is centered in the design of a groupware tool for software quality assessment.

The tool implements the Software Quality Function Deployment (SQFD (Haag et al., 1996; Zultner, 1993)) methodology as the basic approach for evaluating software quality. According to this methodology, software quality is assessed by inspecting a matrix of correlations between a list of technical product specifications and a list of customer requirements. Each cell in this matrix indicates the strength of the relationship between a product specification and a customer requirement using the following numbers: 0, 1, 3 and 9 (Haag et al., 1996).

The approach requires obtaining a consensus view from the customers about the software quality achieved in several milestones established along the software development process. However, this endeavor requires intensive work from the customers, considering they may have conflicting views, the matrix tends to be large and the cell values may have to be individually negotiated. The objective of this groupware tool is to facilitate this negotiation process, supporting several negotiation mechanisms in a same-time, different-place mode.

Figure 3 shows our implementation of the SQFD, using a replicated MS Excel spreadsheet, where the rows represent the customers' requirements and the product specifications appear in the columns. The example shown in Figure 3 was taken from

Herzwurm, et al. (2002). Note that in Figure 3, besides the (0, 1, 3, 9) values, a cell may also be empty and have the following symbols: (?, F, L). These symbols mean respectively that the cell is being negotiated by several customers (?), one customer has a firm position about a correlation (F), and one customer locked the negotiation of a cell (L).

Of course, customers may have several attitudes towards the negotiation of a cell, manifested with suggestions of alternative values, compromising attitudes, and strong positions as well. At the limit, a customer may even decide to block or stall the negotiation process. Therefore, the groupware tool must support this variety of attitudes.

The negotiation process is supported by two main components: the MEG client and the MEG server. The MEG client/server use TCP/IP and RTD (Cornell, 2001) technology to synchronize replicated MS Excel spreadsheets, residing in the users' personal computers, with a centralized data repository. In this architecture, the users do not input cell values directly in the Excel spreadsheets, but in the MEG client. The MEG client interacts with the local replica of the MS Excel spreadsheets and with the MEG server. The MEG server is then responsible for synchronizing the several MEG clients, while RTD is used to synchronize the data on the repository with the Excel spreadsheets. In fact, our case study will focus mostly on the MEG client interface and therefore we do not provide additional details on the system architecture and functionality.

The MEG client implements several user interfaces. Two of them are shown in Figures 4 and 5. In general terms, the MEG user interface is divided in two major areas: the "current situation" area displays the overall status of the negotiation process, reminding about the cell that is currently selected in the SQFD spreadsheet

and showing the different positions taken by all negotiators; while the area below the "current situation" allows the user to express his/her individual position. Both the top and bottom areas change according to the status of the negotiation protocol.

The "current situation" area displays the following information:

- Information about the elements correlated by the cell: "Write emails fast AND in reply include original text for comments"
- Information about the first value specified by a user for the cell: "Main issue: value 1."
- The positions of other users in favor or against this value: "Position: In favor" and "Position: Against." These positions are automatically defined by the system, based on the individual values assigned to the cell by the users (i.e., if the initial value in a cell is 0 and a user afterwards sets a value of 3, then there is one position against the initial value).
- Arguments supporting the positions of users and their corresponding categories: "Security (Risk based arguments)." The arguments and categories are optional and selected from a fixed list defined when the tool is configured for a specific organization.

Both the spreadsheet and MEG briefly explained above can be regarded as shared workspaces. We will now describe the functionality of these shared workspaces using Annotated GOMS. First, let us specify the following objects that exist in the shared workspaces:

cell	One cell of the SQFD spreadsheet for which a value must be agreed by the customers
value	The correlation attributed to the cell
issues	The current status of the negotiation of one cell that is displayed to all negotiators
positions	The component of issues that lists the positions in favor or against the value currently in cell
arguments	The component of positions that lists the arguments supporting a position

The following method describes how a user operates the spreadsheet. The method consists of analyzing the spreadsheet and deciding to propose or to negotiate the value of a cell using MEG. The proposal occurs when the cell is empty (no symbol displayed), while the negotiation occurs when the cell has already a value set (the later case is illustrated in Figure 4, where one user selected 1 and another user is selecting 3). The task is considered finished when the user accepts all values in the spreadsheet.

Method for goal: Negotiate spreadsheet.
Step 1. Accomplish goal: Select cell.
Step 2. Accomplish goal: Analyze situation of cell.
Step 3. Decide: If want to propose value, then
    Accomplish goal: Propose initial value.
Step 4. Decide: If want to negotiate value, then
    Accomplish goal: Negotiate value.
Step 5. Decide: If agreement on all cells, return with goal accomplished.
Step 6. Goto 1.

Next we present two auxiliary methods dedicated to handle cell values. The first one is intended to analyze the cell situation, which includes analyzing feedthrough information about activities of others on the same cell in the spreadsheet. The second method describes proposing the initial value for an empty cell with MEG. This initial value has a special treatment by the users, because all of the subsequently proposed values will be presented by the system as being against or in favor of the first one.

Method for goal: Analyze situation of cell.
Step 1. Verify cell is empty or 0,1,3,9,?,F,L. ← cell modified, cell released, firm released
Step 2. Return with goal accomplished.

Method for goal: Propose initial value.
Step 1. Accomplish goal: Select value from 0,1,3,9. → cell modified
Step 2. Return with goal accomplished.

The following method is dedicated to negotiate a cell value using MEG. The user has several alternative actions while negotiating a value for the cell. Note in step 11 that the system may request a confirmation from the user about the current value of the cell. If all users agree, then the negotiation is considered finished for that cell.

```
Method for goal: Negotiate value.
Step 1. Accomplish goal: Analyze current situation.
Step 2. Decide: If do nothing, return with goal accomplished.
Step 3. Decide: If want other values, then
        Accomplish goal: Propose alternative values.
Step 4. Decide: If insist on a value, then
        Accomplish goal: Support proposed value.
Step 5. Decide: If agree with others, then
        Accomplish goal: Withdraw proposed values.
Step 6. Decide: If change opinion, then
        Accomplish goal: Change proposed values.
Step 7. Decide: If want to block, then
        Accomplish goal: Block negotiation.
Step 8. Decide: If want to unblock, then
        Accomplish goal: Unblock negotiation.
Step 9. Decide: If want firm position, then
        Accomplish goal: Firm position.
Step 10. Decide: If remove firm position, then
        Accomplish goal: Remove firm positions.
Step 11. Decide: If system is requesting value confirmation, then
        Accomplish goal: Confirm value.
        Else goto 1.
Step 12. Return with goal accomplished.
```

Next, we specify several methods related to the analysis of the "current situation." This operation may cascade through several issues, positions and arguments. Observe that MEG does not associate the identity of the users to the positions (except for the user's own position, as shown in Figure 4).

```
Method for goal: Analyze current situation.
Step 1. Accomplish goal: Analyze correlation.
Step 2. Decide: If do not open issues, return with goal accomplished.
Step 3. Verify issues modified. ← issues modified
Step 4. Accomplish goal: Analyze issues.
Step 5. Return with goal accomplished.

Method for goal: Analyze correlation.
Step 1. Verify user requirement and product specification.
Step 2. Return with goal accomplished.

Method for goal: Analyze issues.
Step 1. Accomplish goal: Analyze issue.
Step 2. Decide: If another issue, goto 1.
Step 3. Return with goal accomplished.

Method for goal: Analyze issue.
Step 1. Select issue.
Step 2. Verify lock acquired or firm acquired. ← cell acquired, firm acquired
Step 3. Verify value proposed for cell.
Step 4. Verify positions modified. ← positions modified
Step 5. Decide: If do not open positions, return with goal accomplished.
Step 6. Accomplish goal: Analyze positions.
Step 7. Return with goal accomplished.

Method for goal: Analyze positions.
```

Step 1. Accomplish goal: Analyze position.
Step 2. Decide: If another position, goto 1.
Step 3. Return with goal accomplished.

Method for goal: Analyze position.
Step 1. Select position.
Step 2. Verify in favor or against.
Step 3. Verify arguments modified. ← arguments modified
Step 4. Decide: If do not open arguments, return with goal accomplished.
Step 5. Accomplish goal: Analyze arguments.
Step 6. Return with goal accomplished.

Method for goal: Analyze arguments.
Step 1. Accomplish goal: Analyze argument.
Step 2. Decide: If another argument, goto 1.
Step 3. Return with goal accomplished.

Method for goal: Analyze argument.
Step 1. Select argument.
Step 2. Verify argument type and description.
Step 3. Return with goal accomplished.

The following methods describe the situation when the cell already has a value and the user decides to add, support, withdraw or change values using MEG. Observe that all these actions produce feedthrough information necessary to notify other users about changes in the "current situation."

Method for goal: Propose alternative values.
Step 1. Accomplish goal: Select value from 0,1,3,9. → positions modified
Step 2. Decide: If select another value, goto 1.
Step 3. Return with goal accomplished.

Method for goal: Support proposed value.
Step 1. Accomplish goal: Select argument from list. → arguments modified
Step 2. Decide: If select another argument, goto 1.
Step 3. Return with goal accomplished.

Method for goal: Withdraw values.
Step 1. Accomplish goal: Unselect value from 0,1,3,9. → issues, positions modified
Step 2. Decide: If unselect another value, goto 1.
Step 3. Return with goal accomplished.

Method for goal: Change proposed values.
Step 1. Accomplish goal: Propose alternative values.
Step 2. Accomplish goal: Withdraw values.
Step 3. Return with goal accomplished.

Having described the generic usage of the spreadsheet and MEG, we will now describe two additions to this functionality. The first one addresses the privilege given to a user to block the possibility of reaching a consensus over a cell, a privilege that is common in negotiations and used in various ways to increase individual gains.

Method for goal: Block negotiation.
Step 1. Accomplish goal: Select "block" option. → cell acquired
Step 2. Return with goal accomplished.

Method for goal: Unblock negotiation.
Step 1. Accomplish goal: Unselect "block" option. → cell released
Step 2. Return with goal accomplished.

Another functionality supported by MEG is allowing a user to manifest a "firm" position about a cell value. In this situation, MEG asks the other users if they agree with the firm position, a situation that is shown in Figure 5. If everybody agrees, the negotiation of the cell is considered completed; otherwise, the situation is handled similarly to a blocking situation. Note that the system turns public the identification of a user that manifests a strong position ("JR" in Figure 5).

Method for goal: Firm position.
Step 1. Accomplish goal: Select "firm" option. → firm acquired, who
Step 2. Return with goal accomplished.

Method for goal: Remove firm position.
Step 1. Accomplish goal: Unselect "firm" option. → firm released
Step 2. Return with goal accomplished.

Method for goal: Confirm value.
Step 1. Accomplish goal: Respond "agree" or "not agree". → cell modified
Step 2. Return with goal accomplished.

5.1 Case study Analysis and Obtained Results

The groupware tool analyzed by this case study provides a good example of what we called intensive collaboration, i.e., the whole collaborative task being a repetitive collection of smaller collaborative tasks, since users have to analyze and negotiate a large number of cell correlations to obtain a general consensus. In order to arrive to a successful design solution, we had to make the collaborative tasks around a cell a brief and productive experience. Allowing long and painful negotiations over individual cells would make this an impossible task[1].

Thus, an important goal that we had to accomplish was optimizing the MEG user interface. This was mostly done by working on the "analyze current situation" and "negotiate value" methods. Both of them are complex for several reasons. The "analyze current situation" method is complex because it examines how users perceive the current situation of a cell, which may already have been subject to a long negotiation process and requires the user to recall and go through the correlation, issues, positions and arguments. This requires a significant number of verifications and decisions, with the corresponding cognitive effort.

A solution to this problem, suggested by our analysis, consists of decomposing the task hierarchically. As the method shows, the designed solution places more emphasis on issues (the correlation under negotiation) than on positions (alternative proposals), and more on positions than on arguments. This hierarchical approach affords to focus on the most important information and thus "conserve" cognitive effort.

The "negotiate value" method is complex because of the high number of decisions faced by the user: do nothing, propose, other value, change opinion, etc. Ten

[1] These comments are based on a real-world experiment with the tool, considering a "small" SQFD matrix with 17 product specifications and 9 user requirements negotiated by two pairs of users.

decisions were defined in this method. Nevertheless, we preferred to concentrate all those decisions on this method to optimize the time spent performing this task.

Unquestionably another salient characteristic about the results obtained from the case study analysis is that they do not focus on collaboration as a process. For instance, MEG implements a protocol for handling strong positions with the following steps: (1) a user defines a strong position on a value; (2) the other users are informed and questioned if they accept or not the proposed value; (3) the users respond; (4) MEG collects the responses and, if all agree, then the negotiation of the cell ends, otherwise the cell continues under negotiation but blocked by a user. Although this process may be inferred by a detailed analysis of the described methods, we argue the approach does not make it salient, giving importance to the mediating role of the shared workspaces (spreadsheet and MEG) under the influence of such strong positions. This approach may be beneficial in several circumstances because collaboration supported by groupware is frequently made up of many intermittent and scattered activities. This is clearly what is happening with our case study, where users are free to move between cells at any time, sometimes simply analyzing them for a brief period, other times proposing and negotiating values for a long period. Focusing on the high-level details would hamper the analysis of how users may work in parallel and organize themselves using shared artifacts.

Table 3 provides an outline of how to collaborate using the tool in order to support the previous argument. The table was constructed in the following way. We began by removing all the steps that are not annotated with pre or post conditions from the method descriptions. Then, we also removed the methods that do not have pre or post conditions, while preserving the hierarchy of methods. Finally, we displayed the relationships between the methods handling the same pre and post

conditions. Table 3 shows that users' activities are centered in three shared artifacts: (1) cells; (2) issues, positions and arguments; and (3) firm positions.

A consequence of this focus on shared artifacts is that we had from the beginning to clearly analyze which artifacts were essential to support group work and how they would be used. It was not by chance that we started the case study by specifying the various artifacts available in the shared workspaces. The fact is we would be unable to explain the approach without them.

The approach highlights another interesting aspect for the design of groupware: uncovering the mental conditions needed to accomplish group work. For instance, in the given example, we identified the following mental conditions necessary to handle a conflict: insist on a value, agree with others and change opinion. Of course, we have not tried to specify how individuals define their negotiation strategies. What was interesting to do, from the point of view of a groupware designer, was to associate very clear conditions to specific shared artifacts, in such a way that users may easily grasp the design logic behind each artifact.

In summary, the proposed approach allowed us to:

- Analyze how group work could be organized around shared workspaces and artifacts;
- Analyze the mental conditions required to accomplish work;
- Study different ways to organize the cognitive effort required by collaborative tasks.

6 Benefits and Limitations

Annotated GOMS has been proposed for the case of concerted intensive collaborative work. Moreover, the model excludes face-to-face work. Thus, using the well-known time/space classification of collaborative activities (DeSanctis and Gallupe, 1987), the

model is applicable to distributed synchronous work. The relevant work situations will probably be *scheduled* or *intended*, as opposed to *opportunistic* or *spontaneous*, as classified by Isaacs et al. (1997). The last characterization means interactions are planned in advance or, if they are unplanned, then at least people seek out other people to work together.

The applicability of Annotated GOMS is then restricted to a specific way of collaborative working. Nevertheless, it should be noted that in practice, various ways of working may occur when people associate themselves to generate a joint product. Thus, these people may agree on the basics and then work in parallel, then meet again, then do asynchronous joint work, etc. As a consequence, Annotated GOMS may well be used to model part of this comprehensive enterprise.

The proposed approach affords analyzing in detail the best strategies for organizing intensive collaboration, focusing on mental conditions and cognitive effort, highlighting possible alternatives for working together. One clear difference between singleware and groupware is that in the later case users should not wait for feedforward information as they wait for feedback, and thus strategies have to be devised to accommodate parallel work; however, these strategies also emphasize the need to devise low-level schemes to make users come together from parallel work. The Annotated GOMS addresses exactly the blending of these two design concerns.

Research described in this paper is just a first step in the direction of exploring the inspirations provided by GOMS. Certainly one of the most influential characteristics of GOMS is that it provides quantifiable estimates of human performance, based on experimental measures of time spent by humans executing its operators. Experimental measures for the operators proposed for groupware interaction remain to be done in the future.

Another issue that could be further explored concerns the awareness and coupling processors and devices associated to groupware. In particular, we would like to study how to make awareness output devices more perceptually distinguishable from the output devices.

7 Conclusions

We proposed a discount method to evaluate groupware usability, called Annotated GOMS, which adapts GOMS to the groupware context. Annotated GOMS is based on a cognitive architecture that was specialized from a singleware to a groupware perspective. This new perspective resolves three problems that are not present in the singleware perspective. First, the model addresses the cognitive activities that users must perform to maintain context awareness of collaborative activities, specified as information about who, what, when, how and where are users working on a shared workspace. Second, the cognitive architecture must consider the processors and devices necessary to manage that context awareness. Finally, the cognitive architecture must afford time, space and coupling conditions that are different from the singleware perspective.

The implications for design raised by this model are twofold. As shown in the case study described in the paper, the analysis of collaborative work using Annotated GOMS uncovers the mental conditions necessary to accomplish work, allowing the designers to specify shared artifacts that ease the users' grasping the design logic behind the tool. Designers may also compare different design options based on the analysis of the cognitive workload of a groupware tool.

Acknowledgments

This work was partially funded by Fondecyt (Chile) project No. 1040952.

References

K. Baker, S. Greenberg, and C. Gutwin, "Empirical Development of a Heuristic Evaluation Methodology for Shared Workspace Groupware," in Proceedings of the 2002 ACM conference on Computer Supported Cooperative Work. New Orleans, 2002, pp. 96-105.

H. Beyer and K. Holtzblatt, Contextual Design: Defining Customer-Centered Systems. Morgan Kaufmann, 1998.

S. Card, T. Moran, and A. Newell, "The Keystroke-Level Model for User Performance Time with Interactive Systems," *Communications of the ACM*, vol. 23, no. 7, pp. 396-410, 1980.

S. Card, T. Moran, and A. Newell, The Psychology of Human-Computer Interaction. Hillsdale, NJ: Lawrance Elrbaum, 1983.

J. Carrol, Making use: Scenario-based design of human-computer interactions. The MIT Press, 2000.

P. Cornell, Building a Real-Time Data Server in Excel 2002, Microsoft Corporation, 2001.

G. DeSanctis and B. Gallupe, “A foundation for the study of Group Decision Support Systems”, *Management Science* 33 (5), pp. 589-609, 1987.

P. Dewan and R. Choudhary, "Coupling the User Interfaces of a Multiuser Program," *ACM Transactions on Computer-Human Interaction*, vol. 2, no. 1, pp. 1-39, 1995.

S. Douglas and A. Kirkpatrick, "Model and Representation: The Effect of Visual Feedback on Human Performance in a Color Picker Interface," *ACM Transactions on Graphics*, vol. 18, no. 2, pp. 96-127, 1999.

J. Dyck, C. Gutwin, S. Subramanian, and C. Fedak, "High-Performance Telepointers." Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work. Chicago, Illinois: ACM Press, 2004, pp. 172-181.

J. Fjermestad and S. Hiltz, "An Assessment of Group Support Systems Experimental Research: Methodology and Results," *Journal of Management Information Systems*, vol. 15, no. 3, pp. 7-149, 1999.

C. Gutwin, S. Benford, J. Dyck, M. Fraser, I. Vaghi, and C. Greenhalgh, "Revealing Delay in Collaborative Environments." Proceedings of the 2004 Conference on Human Factors in Computing Systems. Vienna, Austria: ACM Press, 2004, pp. 503-510.

C. Gutwin and S. Greenberg, "The Effects of Workspace Awareness Support on the Usability of Real-Time Distributed Groupware," *ACM Transactions on Computer-Human Interaction*, vol. 6, no. 3, pp. 243-281, 1999.

S. Haag, M. Raja, and L. Schkade, "Quality Function Deployment", *Communications of the ACM*, vol. 39, no. 1, pp. 41-49, 1996.

S. Haynes, S. Purao, and A. Skattebo, "Situating Evaluation in Scenarios of Use." Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work. Chicago, Illinois: ACM Press, 2004, pp. 92-101.

G. Herzwurm, S. Schockert, U. Dowie, and M. Breidung, "Requirements Engineering for Mobile Business Applications." Proceedings of the First International Conference on Mobile Business. Athens, Greece, 2002.

J. Hill and C. Gutwin, "Awareness Support in a Groupware Widget Toolkit." Proceedings of the 2003 International ACM SIGGROUP Conference on Supporting Group Work. Sanibel Island, Florida: ACM Press, 2003, pp. 258-267.

M. Ivory and M. Hearst, "The State of the Art in Automating Usability Evaluation of User Interfaces," *ACM Computing Surveys*, vol. 33, no. 4, pp. 470-516, 2001.

E. Isaacs, S. Whittacker, D. Frohlich, and B. O'Conaill, "Informal communication re-examined: new functions for video in supporting opportunistic encounters". In K. Finn, A. Sellen, and S. Wilbur (eds.): Video-Mediated Communication. Lawrence Erlbaum, New Jerser, NJ, 1997, pp. 459-485.

B. John, "Why GOMS?" *Interactions*, vol. 2, no. 4, pp. 80-89, 1995.

B. John and D. Kieras, "Using GOMS for User Interface Design and Evaluation: Which Technique?" *ACM Transactions on Computer-Human Interaction*, vol. 3, no. 4, 1996.

M. Khalifa, "Computer-Assisted Evaluation of Interface Designs," *The DATA BASE for Advances in Information Systems*, vol. 29, no. 1, pp. 66-881, 1998.

D. Kieras, A guide to GOMS model usability evaluation using NGOMSL. University of Michigan, 1996.

D. Kieras, A guide to GOMS model usability evaluation using GOMSL and GLEAN3. University of Michigan, 1999.

D. Kieras and T. Santoro, "Computational GOMS Modeling of a Complex Team Task: Lessons Learned." Proceedings of the 2004 Conference on Human Factors in Computing Systems. Vienna, Austria: ACM Press, 2004, pp. 97-104.

D. Kieras, S. Wood, and D. Meyer, "Predictive Engineering Models Based on the EPIC Architecture for a Multimodal High-Performance Human-Computer Interaction Task," *ACM Transactions on Computer-Human Interaction*, vol. 4, no. 3, pp. 230-275, 1997.

D. Min, S. Koo, Y. Chung, and B. Kim, "Distributed GOMS: An Extension of GOMS to Group Task." IEEE Conference on Systems, Man and Cybernetics. Tokyo, Japan, 1999, pp. 720-725.

J. Nielsen, "Finding Usability Problems Through Heuristic Evaluation," Proceedings of the SIGCHI conference on Human factors in computing systems. Monterey, California: ACM Press, 1992, pp. 373-380.

J. Nunamaker, R. Briggs, D. Mittleman, D. Vogel, and P. Balthazard, "Lessons from a Dozen Years of Group Support Systems Research: A Discussion of Lab and Field Findings," *Journal of Management Information Systems*, vol. 13, no. 3, pp. 163-207, 1997.

D. Pinelle and C. Gutwin, "Groupware Walkthrough: Adding Context to Groupware Usability Evaluation," in Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves. Minneapolis, Minnesota: ACM Press, 2002, pp. 455-462.

D. Pinelle, C. Gutwin, and S. Greenberg, "Task Analysis for Groupware Usability Evaluation: Modeling Shared-Workspace Tasks with the Mechanics of Collaboration," *ACM Transactions on Computer-Human Interaction*, vol. 10, no. 4, pp. 281-311, 2003.

P. Polson, C. Lewis, J. Rieman, and C. Wharton, "Cognitive Walkthrough: A Method for Theory-Based Evaluation of User Interfaces," *International Journal of Man-Machine Studies*, vol. 36, pp. 741-773, 1992.

S. Rajan, S. Craig, and B. Gholson, "AutoTutor: Incorporating Back-Channel Feedback and Other Human-Like Conversational Behaviors Into an Intelligent Tutoring System," *International Journal of Speech Technology*, vol. 4, pp. 117-126, 2001.

F. Ritter, G. Baxter, G. Jones, and R. Young, "Supporting Cognitive Models as Users," *ACM Transactions on Computer-Human Interaction*, vol. 7, no. 2, pp. 141-173, 2000.

M. Steves, E. Morse, C. Gutwin, and S. Greenberg, "A Comparison of Usage Evaluation and Inspection Methods for Assessing Groupware Usability." Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work. Boulder, CO, 2001.

G. van der Veer and M. van Welie, "Task Based Groupware Design: Putting Theory Into Practice." Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques, DIS ’00. Brooklyn, NY, 2000.

S. Wensveen, J. Djajadiningrat, and C. Overbeeke, "Please Touch Tangible UIs: Interaction Frogger: A Design Framework to Couple Action and Function Through Feedback and Feedforward." Proceedings of the 2004 Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques. Cambridge, MA: ACM Press, August, 2004.

R. Zultner, "TQM for Technical Teams," *Communications of the ACM*, vol. 38, no. 10, 1993.

Type	Format	Sub-type	<X>
Feedthrough	→ <X> , <X> … ← <X> , <X> …	**Basic awareness**	who | what | when | how | where
		Object	object (shared | acquired | released | replicated | modified | moved)
		Mouse	mouse moved
		Key	key (pressed | released)
		Text	text (typed | modified | deleted)
Back-channel feedback			auditory | verbal | visual
Coupling	→ [<X>] ← [<X>]	**Coupling**	(area | objects | events | period)

Table 1 – Syntax of pre and post conditions

Type	Operators
Awareness processor (inputs)	**Verify** (who | what | when | how | where) **Verify object** (shared | acquired | released | replicated | modified | moved) **Verify mouse** (moved) **Verify key** (pressed | released) **Verify text** (typed | modified | deleted)
	Prompt (auditory | verbal | visual)
Coupling processor (outputs)	**Move focus** (area | objects | events | period)

Table 2 – Groupware operators

Propose initial value, Confirm value	→ cell modified →	Negotiate value
Propose alternative values, Withdraw values, Support proposed value	→ issues, positions, arguments modified →	Negotiate value
Block negotiation	→ lock acquired →	Negotiate spreadsheet
Unblock negotiation	→ lock released →	Negotiate value
Firm position	→ firm acquired, basic who →	Negotiate spreadsheet
Remove firm position	→ firm released →	Negotiate value

Table 3 – Outline of collaboration with the tool

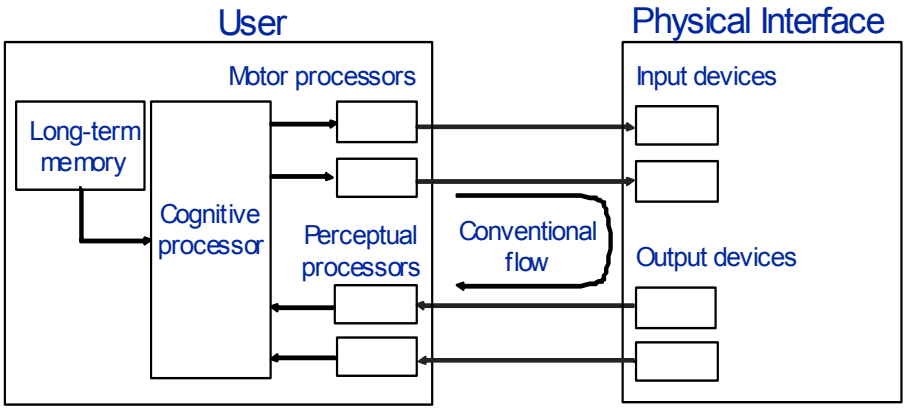

Figure 1 – Singleware architecture

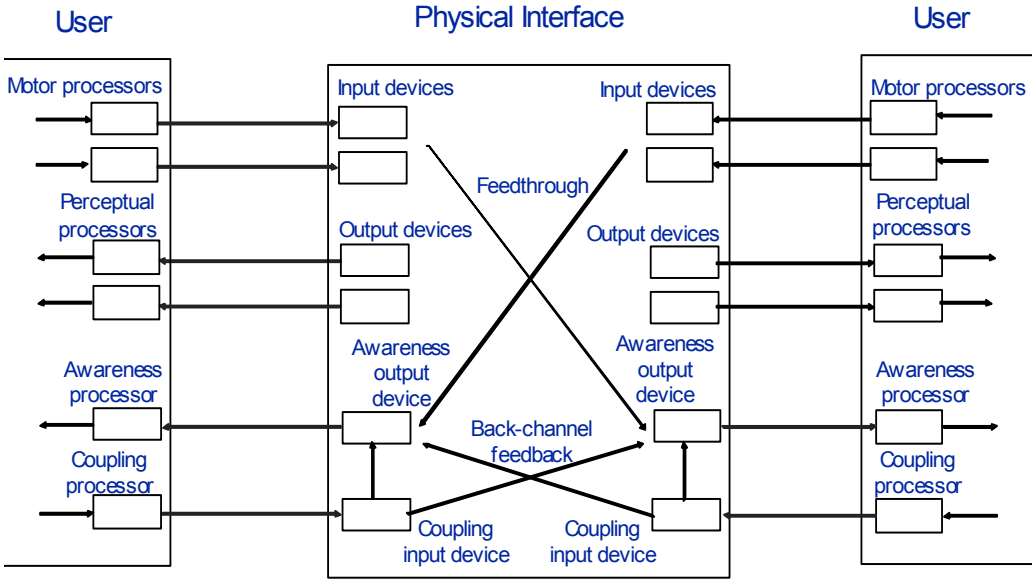

Figure 2 – Groupware architecture

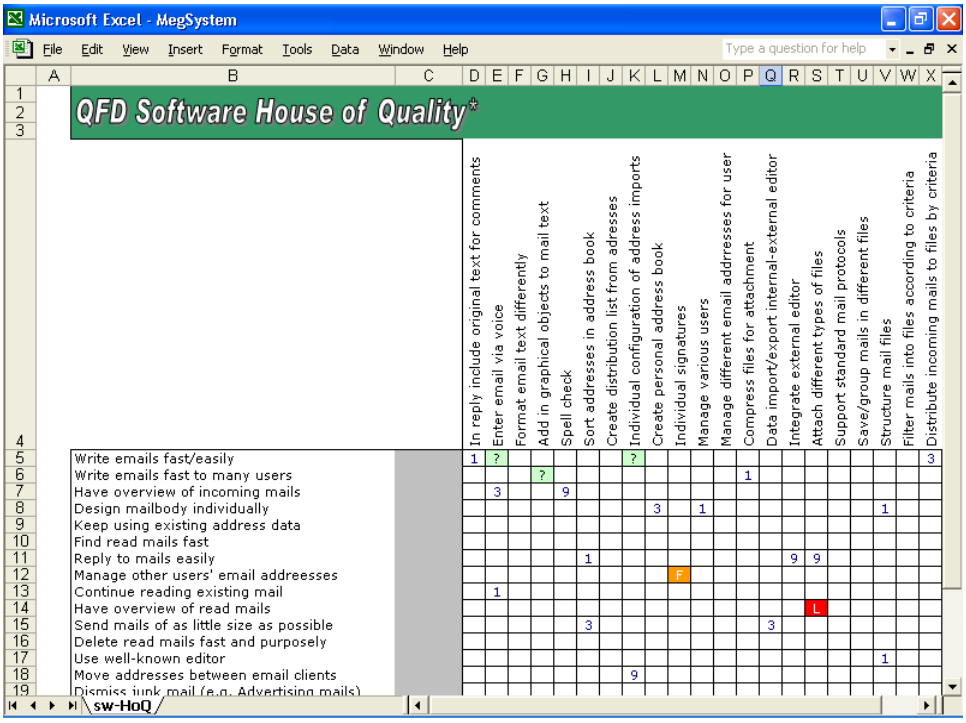

Figure 3 – The SQFD spreadsheet managed by the groupware tool (example from (Herzwurm et al., 2002))

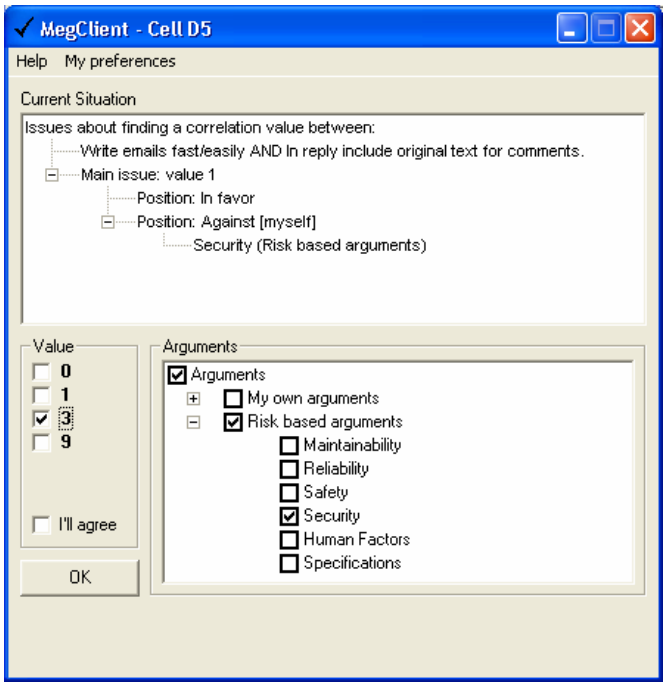

Figure 4 – The MEG user interface

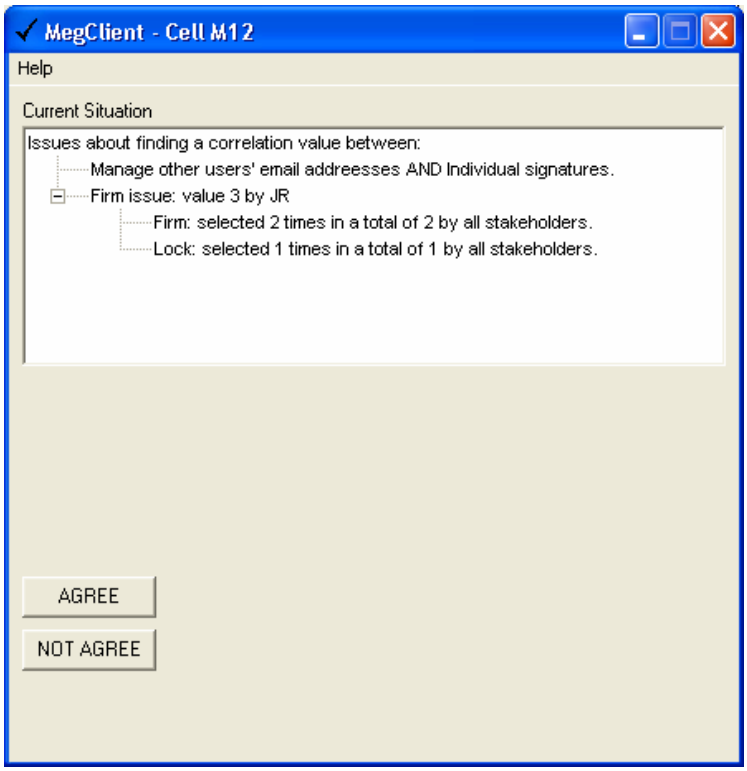

Figure 5 – The “firm” situation in the MEG user interface